1519
Postel

Network Working Group
Request for Comment No. 219
NIC No. 7549
Category:
Updates: None
Obsoletes: None

R. Winter
CCA
3 September 1971

Received 21 Sept 71

USER'S VIEW OF THE DATACOMPUTER

MEMORANDUM

TO: Datacomputer Design File

FROM: R.A. Winter

SUBJECT: User's View of the Datacomputer

DATE: September 3, 1971

Introduction

The datacomputer is a specialized node of the ARPA network that is dedicated to the management of a large, shared database. By large we mean several trillion bits of data, of which at least one trillion are on-line. Shared may mean, for some files, shared by nearly all the users in the ARPA network.

The name, datacomputer, derives from the idea that the system is dedicated to data handling. Though the processor is capable of general computation, it will not be used for that purpose. The processor, like the mass storage device, is only a component of an integrated system, which appears to the user as a black box.

There is one language for addressing the black box: data language. This language defines everything it can do.

All the information presented in this memorandum is about the first of a series of service offerings. We use the term access method to refer collectively to a structure and the operations on it. Being too modest to call the first one AM-1,* we named it DCAM-1.** We expect subsequent DCAMs to generalize DCAM-1. If the need arises, we will design parallel services. All services will use the same data language.

---

* Access Method-1.

** Datacomputer Access Method-1.

System Overview

The users of the datacomputer are programs running on other computers, retrieving data from, and storing it in, the data base. The environments, capabilities, and applications of these programs vary widely; however, a chief design goal is to allow them to share the data.

There is further variation among users in physical connection. Remotely-located users' access is over a narrow link to the datacomputer's low-speed port. Local users are connected to the high-speed port through a link 80 times wider.

Through its ports, the datacomputer accepts two kinds of input: data and requests for services. Data is output through the ports as requested.

In the data base, descriptions are stored separately from the data, and data elements are named, typed, and ordered according to them. A measure of structure independence is obtained by writing access requests in terms of the symbolic names of items in the data description.

Directories are maintained by the system. A hierarchical naming scheme is used, and access controls for privacy and data integrity are provided.

Redundant copies of data and/or journals of changes are maintained by the system and used to effect recovery under system control in case of system error. These facilities can be operated under user control if there is external error.

Since the datacomputer's only interface with the outside world is through its ports, it sees the universe as a group of data streams. Specifically, these are record streams, if one views all transactions (in the data transfer protocol sense) as

records. Associated with each record stream is a data description, allowing the datacomputer to parse the records into named, typed elements.

Thus all data elements--stream elements and data base--are named and fully described. Data type conversion proceeds automatically, as a function of old and new data types, and optional information supplied by the user. Reconfiguration above the element level is a matter of arrangement of elements in records; a full set of capabilities is provided for this. In general, the using program is concerned with the configuration of the stream records that comprise its interface with the datacomputer. The internal configuration of data affects the user only as it limits the data's accessibility or malleability. In fact, the user should not generally have to be aware of the internal data configuration.

Although support on some level for all types of applications is attempted, the first implementation gives particular attention to large, simply-organized, shared files. Emphasis is placed on allowing the user of such files to describe precisely what data is really of interest to him, so that nothing but the desired information is transmitted. This is crucial for avoiding overload of the narrow link, and is accepted as a central design goal.

Data Base Organization

The database contains all information stored in the datacomputer. It is a set of files, which are named, physically distinct, collections of data.

The location of one file, the file directory, is known to the system. It contains the names, locations, and certain attributes of all the other files. Access to this file is restricted.

Internally, each file has its own organization, but each organization is a particular application of a general model. The particular

application is defined by a file description associated with the file.

In the general model, each file contains uniquely numbered records. Each record contains named fields. A field of a certain name may occur more than once in a given record, and a unique number is associated with each occurrence. A field contains an elementary piece of data, the value of the field.

The records are variable in format and size. Fields are variable in length.

In addition to the records themselves, each file can contain an index. The system maintains the index to the specifications of the user. Conceptually, the index contains lists of pointers to records having certain properties. A typical list might point to the records containing the field STATE with the value MASSACHUSETTS.

The system supplies a unique, permanent, identifier for each record. This identifier maps trivially into a location in the file, or at worst, into a small region in which the record can be quickly located. The identifier is used in pointers to the record, both from the index and from other records.

Besides the physical ordering, defined by record location, a logical ordering will be maintained on request by the system. This can be based on some simple function of record contents, such as the value of certain fields. Alternatively, the user can compute the function, and simply supply the result (for example, by saying "insert this record after that one"). Retrieval from such ordered files can be made either in physical or logical order.

In all such ordered files, if insertions are made, space must be reserved for them and garbage collection must be done periodically.

A single field value is viewed as a homogeneous string of characters or basic data units. It is described by giving the type (e.g., ASCII, BIT, binary integer, etc.) and the length in some unit associated with the type. When the length of a field is constant throughout the file, it is stored in the file description; otherwise it appears with each occurrence of the field. The type of a field is constant.

The information in the file description is sufficient to parse a record into (field name, value) pairs. Also, given such a set of pairs, and a file description, the system can produce a record satisfying the description. Mapping in either direction, there is only one possible result.

With a record, a file description, and a (field name, value) pair to store in the record, there is also only one new record that can result.

Thus a file description defines all the possible formats for a record from a particular file.

Stream Organization

Streams are sequences of records passed from using programs to the datacomputer or vice versa. The format of the records is defined as in the file description. Thus streams have the same organization as files, except they cannot be indexed. The operations defined on streams are more limited than those defined on files, since the records must be accessed in sequence.

There is no concept of permanent storage for streams. The records move past the datacomputer one at a time, as though they were on a conveyor belt.

One record, the current record, is available to the datacomputer in each stream. To begin formatting the subsequent record in an output stream, the datacomputer transmits the current record. To access the next record in an input stream, the datacomputer relinquishes access to the current one.

Operations

When the user is interested in the contents of his whole file in solving the problem at hand, the datacomputer's job is simple in terms of information retrieval. There may be reformatting or reordering, but location of the right data to operate on is trivial. However, this will not be the standard usage of the datacomputer, particularly for the remote user.

For most problems, the datacomputer expects to subset the file before doing anything else. The larger the file compared to the subset, the less acceptable it is to transact with the full file in order to form the subset. And the datacomputer will have such enormous files that using anything but a very small subset in one problem is most unusual. Thus, subsetting without examining the entire file is a fundamental requirement.

Normally, the subset will be considered formed when a list of the relevant record id's or record addresses is known.

The index of the datacomputer file can be thought of as a collection of primitive record id lists that the file designer expected to be useful in forming interesting subsets. The values of all important fields can be indexed. For example, every word in a field containing a string of text might be indexed. In fact, an arbitrary function of the contents of the record, or the relation of the record to other records can be indexed.

The common logical operators (AND, OR and NOT) are defined for record subsets. Arbitrarily complex expressions of them can be evaluated with relatively little processor time or I/O. The ease of this operation results from careful design of the index and strategies, the most important of which is the parallel evaluation of the Boolean functions on large groups of records.

Certain statistical operations--like counting the number of records satisfying a certain Boolean condition--can be done directly on the index. This can be used to derive question-answering strategies heuristically, or can be the direct input to a statistical study.

Once the index has done all it can in subsetting, attention turns to the records themselves. Certain conditions cannot be evaluated using the index; an obvious case is the selection of records based on the value of an unindexed field. Also, certain data structures cannot be explicitly represented in the file:record:field model. These must be constructed by the user, out of groups of records linked by pointers, or using other special mechanisms. The class of operations that is useful in further record selection consists of field content testing, pointer chasing, simple computation in the numerical and symbolic senses, and various operations below the data element level, such as pattern matching, string manipulation, etc. Such operations require a control structure approaching that of the general purpose higher level language. It is our intention to make all of this available, though not with the goal of providing a computation facility, but rather, a data management facility that is capable of using as much knowledge as the programmer can supply.

A simple set of primitives is required for file maintenance in the data structure we are talking about. The operations are:

1. add a field/record
2. delete a field/record
3. replace a field/record.

The difficult part, as in retrieval, is locating the element to be operated on. Notice that individual record formats can be changed at will: the set of possible formats is limited only by the file description.

When record contents are changed, index entries that are a function of them must be changed also. When the function determining what is to be indexed is part of the file description, the maintenance of the index is automatically performed by the system. Otherwise, this is the responsibility of the user.

All fields in a record can be optional, variable length, allowed to occur an arbitrary number of times (up to some fixed limit for each field). Fields can be present and later be deleted from any record. Fields can be added to the file description at any time. The only reason for limiting the flexibility of a record format is to reduce storage overhead.

Applications

The system outlined here is intended to be suitable for many applications; some examples are:

1. Storage and retrieval of dumps and other unstructured files. The system will happily pack away your one enormous record, as quickly and painlessly as possible.

2. Applications that would normally be set up on tape: sequentially accessed files that are copied over when they are changed. Most record formats should be able to remain just as they are. If you want to operate this way, the datacomputer imposes no overhead (such as indexing) on you. The datacomputer willingly acts as unsophisticated as a tape drive; it will pass your file, adding and changing records as it copies them. It will pull off the interesting ones, reconfigure if desired, and transmit them to you. When you describe the data, you have solved the data sharing problem for this application.

3. Simple-minded direct access applications. The great hairy index structure neatly degenerates to imitate indexed sequential, simple directly-addressed files, and other old standbys in the direct access world.

4. Text/document retrieval. The indexing is made for this kind of applications. In addition, documents can point to sub-documents, related documents, etc.

5. Content-oriented, rapid retrieval applications are the specialty of the house.

6. Large data bases used for statistical analysis or modeling such as the census, the common social science data bases, etc.

7. Applications in which data element groups (such as records) are related in a complex fashion, and the intelligence of the datacomputer, which is close to the data and remote from the computational facility, can be put to good use.

In all of these, an important consideration is size. We hope to handle these applications properly on the datacomputer, even when the files are of extraordinary size.